Videojuegos

Curso de Diseño y Programación

Nº 3 **5,99** euros

*Diseño de los **elementos** que componen un juego*

***Milkshape3D:** Modelado en baja poligonización*



*Herramientas para crear el **sonido** de un juego*

3



Videojuegos

Curso práctico de Videojuegos

AUTOR DE LA OBRA
Marcos Medina
DIRECCIÓN EDITORIAL
Eduardo Toribio
etoribio@iberprensa.com
COORDINACIÓN EDITORIAL
Eva-Margarita García
eva@iberprensa.com
DISEÑO Y MAQUETACIÓN
Antonio Gª Tomé
PRODUCCIÓN
Marisa Cogorro
SUSCRIPCIONES
Tel: 91 628 02 03
Fax: 91 628 09 35
suscripciones@iberprensa.com

FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: Gráficas Don Bosco
DUPLICACIÓN CD.ROM: M.P.O.
DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Iberprensa
www.iberprensa.com

CONSEJERO
Carlos Peropadre

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, 7 (Pol. Ind. "El Nogal")
28110 Algete (Madrid)
Tel.: 91 628 02 03
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-35934-2002
ISBN: Coleccionable: 84 932417 2 5
Tomo 1: 84 932417 3 3
Obra Completa: 84 932417 5 X

Aprende divirtiéndote

Bienvenidos a **Programación y Diseño de Videojuegos**, la primera obra coleccionable cuyo objetivo es formar al alumno en las principales técnicas relacionadas en el desarrollo completo de un videojuego.

A lo largo de la obra el lector aprenderá programación a nivel general y a nivel específico con ciertas herramientas y lenguajes, aprenderá a trabajar con aplicaciones de retoque de imagen y también de diseño 3D y animación. Descubrirá las aplicaciones profesionales más importantes de audio y conocerá la historia de lo que se denomina "la industria del videojuego", los últimos 20 años, los juegos que marcaron un avance, sus creadores y en general la evolución del videojuego.

Pero además, esta obra tiene un segundo objetivo, desarrollar y potenciar la creatividad del lector, nosotros a lo largo de las diferentes entregas pondremos las bases y tú pondrás tu ingenio, tu creatividad y tu capacidad de mejorar.

Comienza aquí un viaje de 20 semanas articulado en 400 páginas y 20 CD-ROMs cuya finalidad es proporcionar las bases mínimas para después cada uno continuar su camino.

Recuerda que para alcanzar el éxito necesistas cumplir tres condiciones: que te gusten los juegos, poseer cierta dosis de creatividad y finalmente capacidad de estudio.

Una la cumples seguro.

sumario



PARA ENCUADERNAR LA OBRA:

- *Para encuadernar los dos volúmenes que componen la obra "Programación y Diseño de Videojuegos" se pondrán a la venta las tapas 1 y 2.*
- ***Los suscriptores recibirán las tapas en su domicilio sin cargo alguno como obsequio de Iberprensa.***

SERVICIO TÉCNICO:
Para consultas, dudas técnicas y reclamaciones Iberprensa ofrece la siguiente dirección de correo electrónico: games@iberprensa.com

PETICIÓN DE NÚMEROS ATRASADOS:
El envío de números sueltos o atrasados se realizará contra reembolso del precio de venta al público más el coste de los gastos de envío. Pueden ser solicitados en el teléfono de atención al cliente 91 628 02 03

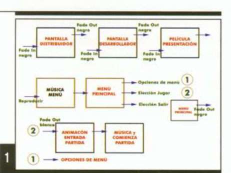
Diseño de los elementos I: desarrollar la **idea**

En el número anterior definimos, de una forma práctica, todos los aspectos de nuestro juego. Elegimos un tipo de juego de acción en 3D con varias vistas de cámaras seleccionables por el jugador, un estilo gráfico algo futurista y una ambientación sonora con bastante ritmo. El juego tendrá posibilidad multijugador y la jugabilidad se centrará en el combate con otros adversarios en un terreno acotado. Pues ya podemos ponernos manos a la obra.

(■) EL GUIÓN

Básicamente hay dos tipos de guiones: el literario y el técnico. Por ejemplo, una aventura gráfica precisa de los dos: el técnico para describir las facetas gráficas, sonoras o de programación, y el literario para contar el desarrollo de la aventura (descripción de personajes, diálogos o situaciones). Los guiones varían según el tipo de juego y para explicar cada una de estas variaciones necesitaríamos muchas páginas. Por ello, nos centraremos en el tipo de juego que hemos elegido. Generalmente, en un videojuego de nuestras características –donde la acción se centra en el combate entre jugadores y en un único tipo de situación– no existen diálogos entre personajes ni cambios para seguir una historia. Básicamente, esta aventura la fabrica el mismo jugador con su actuación o elección en el menú principal. Precisamente, de él depende si se combate en un terreno arenoso o en un medio acuático, o si se compite con otro jugador humano o con el ordenador. Por lo tanto, realizaremos un guión técnico.

Este guión servirá para conducirnos a través del desarrollo y, fundamentalmente, se trata de una especie de cuaderno de notas en el que describiremos todas las facetas del juego de manera secuencial. Es decir, en qué orden y qué elementos aparecen en nuestro juego desde que empieza su ejecución. Podemos ayudarnos con métodos como, por ejemplo, un esquema gráfico como se muestra en la figura 1, donde observamos nuestro guión esquematizado, describiendo cada secuencia técnica dentro de cuadros.



En la figura se muestra un esquema gráfico del guión de nuestro juego.

(■) GUIÓN TÉCNICO DE *ZONE OF FIGHTERS*

Comenzamos el guión describiendo paso a paso la ejecución del juego:

- **1.** Aparece pantalla de presentación del distribuidor desde fondo negro (FADE IN –fundido de entrada– a negro)
- **2.** Desaparece pantalla del distribuidor hacia fondo negro (FADE OUT –fundido de salida– a negro)
- **3.** Aparece pantalla del desarrollador desde fondo negro (FADE IN –fundido de entrada– a negro)
- **4.** Desaparece pantalla del desarrollador hacia fondo negro (FADE OUT –fundido de salida– a negro)
- **5.** Aparece película de presentación (FADE IN negro)
- **6.** Termina película de presentación
- **7.** Inmediatamente activamos música del menú principal
- **8.** Inmediatamente aparece menú principal
- **9.** Esperar elección del jugador
- **10.** En caso de elección del jugador cambiar de menú desplegando las opciones
- **11.** En caso de elección de salida del juego FADE OUT a negro de la pantalla del menú
- **12.** En caso de comenzar partida; FADE OUT a blanco de la pantalla del menú principal
- **13.** Comienza la animación de entrada a una nueva partida
- **14.** Activar música de juego
- **15.** Comienza partida
- **16.** Si el jugador elige opciones durante la partida se para la acción y aparece menú de opciones.
- **17.** Si el jugador decide salir de la partida, FADE OUT a blanco y aparece menú principal con FADE IN desde blanco.
- **18.** Si muere el jugador, termina la partida. Aparece pantalla o acción de GAME OVER y FADE OUT a blanco. Aparece el menú principal con FADE IN desde blanco.

> **NOTA**
>
> **Lo importante es que los encargados de interpretar el guión deben entenderlo fácilmente para poder desarrollar correctamente el diseño.**

Esto es básicamente una forma de realizar un guión técnico.

(■) BOCETOS Y *STORY BOARD*

Los bocetos representan la antesala del aspecto del juego. Son dibujos, más o menos artísticos, que plasman la idea que se tiene de los gráficos. Generalmente, se suele dibujar a los protagonistas, los decorados y sus elementos como construcciones, vegetación, etc. Sin embargo, siempre es muy conveniente tener al menos algunos trazos de otros aspectos como los menús, iconos, indicadores de pantallas y cosas así.

No vamos a enseñar en esta obra a dibujar bocetos, pero sí daremos una orientación de cómo deben ser. No es importante que el dibujo sea una auténtica obra de arte, pero sí que la idea que se quiere transmitir sea reconocida perfectamente por los grafistas. Los bocetos suelen estar en blanco y negro cuando se realizan para el modelado posterior. El uso del color será importante y necesario cuando el diseñador, por ejemplo, describe un tipo de ambiente especial y es imprescindible reflejar el colorido. También sería útil el color, por no decir fundamental, para el grafista 2D que realiza las texturas de los modelos; es primordial que éste sepa si el casco de la nave es gris metálico, si la piel del personaje principal es oscura o si el terreno es de tierra roja o amarilla para poder realizar su trabajo.



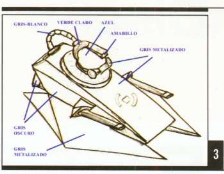
Ejemplo de boceto coloreado, aunque los bocetos suelen ser en blanco y negro.

No obstante, es suficiente con indicar los colores por escrito sobre el boceto en blanco y negro, ya que la misión de estos dibujos es servir de guía para el modelador y el texturizador.

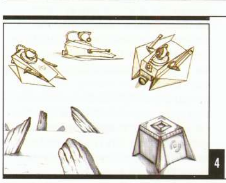
Otra regla básica a tener en cuenta es desde qué ángulo se ve el dibujo en el papel. Todo depende de qué queremos representar. Es conveniente dibujar a los protagonistas del juego en un mínimo de tres vistas, en las que se puedan distinguir la parte frontal, lateral y trasera del modelo. Si lo que queremos es dibujar algún tipo de construcción, de formas más o menos geométricas, basta con una perspectiva isométrica en la que se vean al menos dos caras. Para ambientes y elementos de decoración (piedras, vegetación, etc.) es suficiente con una vista lateral.

El *storyboard* es similar a un guión técnico esquematizado, pero con la salvedad de que incluye viñetas dibujadas con anotaciones de texto. Se utiliza para ilustrar las fases que tendrá un anuncio, una película o, en nuestro caso, una animación. Las viñetas contienen el desarrollo de una y el desarrollo de ésta se refleja con textos técnicos que indican movimientos de cámara o de modelos e incluso cambios del entorno. El *storyboard* ayuda a los grafistas en el modelado y la animación.

En *Zone of Fighters* tendremos una animación al comienzo de cada nueva partida, consistente en la entrada de las bionaves en el interior de las urnas que contienen los terrenos de combate. En la figura 5 podemos ver el *story board* completo de la animación. En este caso, el sistema escogido es completamente gráfico. Cada viñeta se divide en dos partes: el dibujo de la izquierda, más detallado, representa cómo empezaría la acción y el de la derecha, más pequeño y esquemático, el desarrollo de esta acción mediante flechas.



Indicaciones por escrito del color sobre un boceto en blanco y negro.



Ejemplos de bocetos: es conveniente tener varias vistas de cada modelo.

(■) LOS PROTAGONISTAS DEL JUEGO

Entendemos por "protagonistas" a todos los personajes que intervienen en el juego. El papel del jugador lo interpretará el protagonista principal, ya sea un guerrero con espada, un jugador de fútbol, etc. De todas formas, en muchos casos, nuestro protagonista puede quedar oculto en el interior de un avión comercial o, en nuestro caso, de una nave de combate. Los demás protagonistas serán los que, de alguna u otra manera, intervienen en la historia del juego, bien como enemigos directos en la acción o como extras. Básicamente, estos personajes dan sentido a la historia del juego, por lo que no hace falta comentar que un diseño correcto es fundamental para dar credibilidad a la historia.

(■) PERSONAJE PRINCIPAL

Más del 50 por ciento del éxito de un juego se debe al protago-

nista principal. Es con quien toma contacto y se identifica el jugador en todo momento. Un claro ejemplo de esta importancia la encontramos en los RPGs (ROL). En este tipo de juegos es trascendental un sistema de configuración del protagonista principal por parte del jugador, ya que el desarrollo de la historia del juego depende en gran medida de este hecho. Actualmente, raro es el juego que no permita una configuración personal del protagonista principal; desde asignarle el nombre del jugador hasta cambiarle el aspecto físico.

Con la llegada de los FPS o juegos en primera persona el jugador se aproxima al máximo al personaje principal, ya que se convierte en él mismo y ve a través de sus ojos, debido a la perspectiva subjetiva que se tiene de la acción. La llegada del sistema de cámara en tercera persona –en parte originado por la necesidad que el jugador demanda de verse a sí mismo representado por un héroe– contribuyó a mejorar la calidad del diseño de este elemento esencial en todo videojuego. Aunque, generalmente, es posible cambiar de vista en un juego en 3D, es la tercera persona, sin duda, la preferida por los jugadores; tanto que incluso los FPS han adoptado esta opción. Si hacemos una pequeña observación, el porcentaje de usuarios de juegos de acción de sexo femenino aumentó considerablemente tras la salida al mercado del juego *Tomb Raider*, sólo por el hecho de que la protagonista principal era una mujer.

5

Ejemplo de Story Board completo de la animación.

Cada vez son más los tipos de acciones que el personaje principal puede realizar, con una calidad de movimientos pasmosa y una apariencia aún más real. Este aspecto del diseño es importantísimo para asegurar el éxito de un juego. Pensemos un momento por qué triunfó mundialmente el juego *Prince of Persia* entre los primeros juegos para ordenador. Aparte de tener una ambientación gráfica estupenda, el personaje principal estaba dotado de unos movimientos asombrosos. A veces el desarrollo del juego se hacía pesado y monótono, pero el simple hecho de ver cómo tu personaje se movía era suficiente para tenerte horas delante de la pantalla.

Puede ocurrir que, aunque nuestro protagonista sea humano, va conduciendo algún tipo de vehículo; por ello, el jugador sólo ve dicho artefacto. Entonces el diseño del protagonista principal se centra en el vehículo y no en el ser antropomorfo.

En *Zone of Fighters* nuestro protagonista maneja una bionave de combate; si observamos en la figura 2 el diseño de este vehículo, vemos cómo parte de nuestro personaje asoma de la nave; por lo tanto, hemos conseguido una representación más cercana del jugador.



6

La configuración del protagonista principal ayuda más a que el jugador se identifique con él.

(■) OTROS PERSONAJES

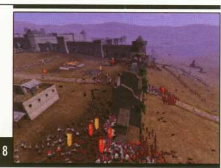
A este sector pertenecen todas las demás criaturas o artefactos que comparten historia con el personaje principal. Realmente son los que dan la razón de ser al juego. Sin ellos no tendríamos acción, ni siquiera una historia que jugar. Hay muchas clases de estos personajes, dependiendo de su papel. Los tenemos protagonizando la historia codo con codo con nuestro intérprete o bien de segundones en papeles de relleno. Pero, sin lugar a dudas, son los elementos más difíciles y gratificantes de programar en un juego. Además de un diseño gráfico, hay que sumarle un diseño de

> **NOTA**
>
> **Los personajes deben ser capaces de mostrar cierta inteligencia, para que así el juego sea más interesante.**



7

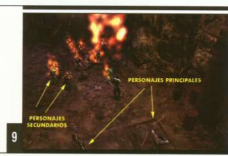
Price of Persia fue de los primeros juegos en dotar a los personajes de movimientos realistas.



En los juegos de estrategias, la Inteligencia Artificial de los personajes que controla el ordenador posee un nivel muy elevado de realismo.

comportamiento. Implementar cierta inteligencia a estos personajes constituye, hoy día, uno de los pilares tecnológicos más importantes en el desarrollo de videojuegos. Si no, que se lo digan a Iwatani, el creador del *Pacman*, que la mitad del tiempo que duró su desarrollo lo gastó en diseñar el comportamiento de los fantasmas del juego, que fue –sin lugar a dudas– lo que mantuvo pegado al juego a millones de personas.

Toma mayor importancia, en los días que corren, la IA (Inteligencia Artificial) de los personajes secundarios, debido a la jugabilidad que demanda el jugador, cada vez más exigente. Un ejemplo bien claro de lo imprescindible que se vuelve el diseño de la IA de personajes lo encontramos en los juegos de estrategia, donde la jugabilidad sólo existe si las unidades enemigas dan la sensación de estar comandadas por otro jugador. El ansia de superación invita al usuario a seguir jugando y, para que eso ocurra, el comportamiento de los enemigos tiene



El comportamiento de los personajes secundarios de un juego tiene que parecer lo más real posible para ganar en jugabilidad.

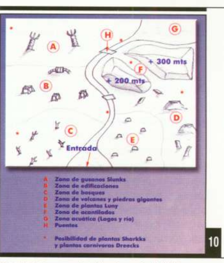
que parecer imprevisible y con cierto sentido de competición. Los comportamientos fijos y, por consiguiente, predecibles, provocan aburrimiento en la acción.

El diseño de los personajes secundarios depende, en gran medida, del tipo de juego al que pertenecen. En juegos tipo Arena o juegos de combate como *Quake III* o *Unreal Tournament*, ocurre que los personajes secundarios pueden ser otros jugadores conectados a la partida por cable o Internet. *Zone of Fighters* es un juego de este tipo, pero con el añadido de otros personajes que viven en el campo de batalla y que, forzosamente, forman parte de la acción entre los jugadores. Así, nos encontramos con animales y plantas carnívoras letales que ayudan a añadir aún más emoción a la partida. La naturaleza inesperada de la situación de estas criaturas en el terreno de combate puede dar una sensación de comportamiento imprevisible, aunque en realidad no lo sea, debido a que la atención del jugador está desviada por el verdadero núcleo de la acción, el combate entre las bio-naves.

(■) EL MAPEADO: LAS ZONAS DE COMBATE

Todo juego transcurre en un entorno definido. Podemos viajar por el espacio, recorrer una tierra imaginaria o luchar en el interior de cavernas, pero todas estas situaciones deben estar previamente diseñadas. Al igual que se dibuja el mapa de un país o una región, es necesario hacer lo mismo en nuestro juego.

En el mapeado, además de dibujar las distintas zonas, niveles o habitaciones por donde transcurrirá el juego, hay que indicar qué decoración, seres o vegetación se pueden encontrar, así como todos aquellos objetos especiales que pueda hallar el jugador como armas, munición, llaves, bonos, etc.



Boceto del mapeado del primer nivel del juego Zone of Fighters.

Como ya sabemos, *Zone of Fighters* transcurre en grandes superficies de terreno con bosques, ríos, volcanes, edificios, animales, etc. Igualmente, habrá repartidos distintos objetos que pueden recoger los jugadores. Entonces, para que el grafista y el programador sepan cómo fabricar y colocar todos estos elementos, es necesario hacer un croquis o dibujo de los terrenos indicando el lugar exacto donde se ubicarán.

Hay casos en que un objeto o animal pueden aparecer en cualquier parte del mapeado aleatoriamente; por ejemplo, los bonos de puntos y la munición o el ataque de un gusano. En ese caso, basta con indicarlo o marcar la zona de la posible situación. En la figura 9, podemos ver el mapeado de la primera zona de combate de nuestro juego con una leyenda explicativa.

En el próximo número...

... definiremos nuestra bionave de combate: aspecto, acciones, armas, etc., así como todos los demás personajes del juego. Estudiaremos sus comportamientos y diseñaremos su forma.

Modelado en baja poligonización con Milkshape3D (I)

Modelar en baja poligonización consiste en construir modelos en 3D con pocos polígonos. El polígono más pequeño que podemos formar es el triángulo. Éste, a su vez, está formado por tres vértices unidos entre sí. Este sistema es básico para el modelado de objetos para videojuegos, ya que de la cantidad de polígonos que estos tengan dependerá el rendimiento del juego.

Para modelar en baja podemos utilizar varios procedimientos. El más sencillo es crear objetos por medio de primitivas, como cajas, cilindros o esferas, editando luego sus vértices. Sin embargo, aunque este sistema no permite modelar objetos complejos -como formas orgánicas- sin utilizar gran cantidad de polígonos, es muy útil para fabricar otros más sencillos como edificios, vehículos, etc. El siguiente sistema es más laborioso, pero también más flexible. Permite modelar cualquier tipo de objeto ya sea orgánico o inorgánico y consiste en ir formando las caras del objeto utilizando triángulos, los cuales creamos previamente uniendo vértices.

EMPEZAR CON MILKSHAPE 3D

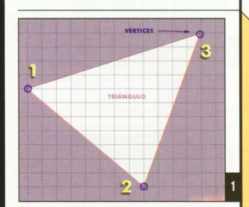
Milkshape3D surgió de la mano de los creadores del *HalfLife* para modelar los objetos que luego utilizarían en su juego. Posteriormente, se abrió al público como aplicación shareware y, debido a su coste más que asequible, es el más usado por los grafistas *freelance*. Es fácil de usar y soporta todos los formatos conocidos para juegos. Es tremendamente popular en la comunidad de desarrolladores de *Blitz 3D*, tanto que incluso existe una utilidad gratuita que se instala en el *Milkshape 3D* (plugin) para exportar los modelos al formato gráfico del *Blitz 3D* (*.B3D*). *Milkshape 3D* es básicamente un modelador de baja poligonización que además permite crear esqueletos y dotar de animación a los modelos. Aun así, nosotros nos centraremos sólo en el modelado, ya que la animación de nuestros objetos la realizaremos con otra aplicación más especializada.

Al ejecutar el programa, podemos ver que está dividido en distintas zonas. El área de vistas, la zona de edición, la ventana de mensajes y el Keyframer para la animación.

ÁREAS DE VISTAS

Esta zona es similar a cualquier otra aplicación 3D. Tenemos cuatro ventanas, las cuales muestran una vista distinta del objeto (vista lateral, frontal, trasera y perspectiva). En "Viewports" situado en el menú "Windows", podemos cambiar la disposición y el número de estas vistas. Pulsando el botón derecho del ratón sobre cualquiera de las vistas, abriremos un menú con todas las opciones de visualizado, que desde este momento llamaremos "Menú Emergente". Eligiendo la opción "Maximize", la ventana de visión sobre la que estamos ocupará toda la pantalla. Para volver a la configuración de vistas anterior, basta con elegir de nuevo "Maximize".

Antes de continuar, vamos a ocultar el Keyframer y la ventana de mensajes, ya que no los vamos a utilizar. Para ello, deseleccionamos en el menú



1

Podemos modelar los objetos usando triángulos, los cuales creamos uniendo vértices.



2

En las ventanas de vistas se nos muestran distintas perspectivas del objeto.

NOTA

Para acceder al menú emergente de opciones, es suficiente con pulsar el botón derecho del ratón sobre cualquiera de las vistas.

3

En la zona de edición están todas las herramientas necesarias.



4

Creamos un cubo arrastrando el ratón. El color rojo indica que ya es un objeto.

"Window" las opciones "Show Keyframer" y "Show Message Window". Observamos que en la parte superior de cada vista hay un encabezamiento (caption) con cuatro casillas. La primera es un pequeño menú desplegable que permite cambiar la vista en esa ventana, la segunda casilla nos indica el tamaño del *grid* o parrilla modificando el zoom de la cámara, y la tercera y cuarta casilla indican respectivamente lo cerca y lejos que puede llegar el campo de visión de la cámara. Nosotros vamos a utilizar los valores estándar en las vistas lateral, frontal y trasera, (4, -1024, 1024) y para la vista 3D cambiaremos a 50, 0.1 y 4096.

Los encabezamientos se pueden ocultar deseleccionando en el menú "Window" la opción "Show Viewport Caption".

Observamos también que en el centro de cada ventana de vistas hay tres líneas de colores. Estas líneas representan los ejes X (de color amarillo), Y (de color azul) y Z (de color rosa). Estos ejes se pueden ocultar deseleccionando "Show Axis" en el "Menú Emergente". A continuación vamos a situarnos en la vista 3D. Para poder rotar la cámara, movemos el ratón manteniendo pulsado el botón izquierdo. Para desplazarla sobre la vista, haremos lo mismo con el ratón, pero pulsando la tecla *CTRL*, y para acercar o alejar la cámara dejaremos pulsado el botón izquierdo más la tecla *SHIFT* pero moviendo el ratón hacia arriba o abajo.

Todos estos procedimientos son aplicables al resto de las ventanas de vistas, a excepción del zoom de cámara que sólo es posible si está activada la opción "Move", la cual explicaremos más adelante.

ZONA DE EDICIÓN

Es aquí donde se encuentran todas las herramientas necesarias para trabajar con *Milkshape 3D*. La mayoría de las funciones más utilizadas para modelar y editar los objetos van asociadas a una tecla de función. Es muy conveniente tener esto bien claro para trabajar mucho más cómodamente. La zona de edición está compuesta por cuatro grupos de funciones: "Model" (modelado), "Groups"(grupos), "Materials"(materiales) y "Joints"(uniones).

En "Model" tenemos todas las herramientas necesarias para construir y modificar los objetos 3D, desde simples vértices hasta primitivas completas. En "Groups" encontramos lo necesario para seleccionar y agrupar objetos de un modelo. En "Materials" podemos cargar, modificar y asignar los materiales que luego irán en cada objeto. Y por último, en "Joints" se engloba todo lo necesario para crear esqueletos mediante la unión de huesos.

CREAR, SELECCIONAR UNA PRIMITIVA

Para familiarizarnos con las opciones de edición básicas del *Milkshape 3D*, vamos a utilizar primitivas.

En primer lugar, vamos a crear un cubo en medio de la cuadrícula. Seleccionamos la pestaña "Model" de la zona de edición. Pulsamos sobre el botón "Box". A continuación, nos colocamos en la vista frontal (front) y sin dejar de pulsar el botón izquierdo del ratón nos desplazamos. Observamos cómo se va formando un cubo en todas las vistas. Al soltar el botón, el cubo se vuelve de color rojo indicando que ya es un nuevo objeto. Si volvemos a pulsar el botón y desplazamos el ratón, estaremos creando otro cubo nuevo y el primero se vuelve de color blanco. (ver fig. 4).

Como pequeño ejercicio y aprovechando que hay un objeto como referencia, podéis practicar el desplazamiento de la cámara en la vista 3D. Cuando hayáis practicado suficiente probad en las demás vistas. Si en algún momento perdéis el panorama el cubo, se

NOTA

En cada vista, la línea de color amarillo representa el eje X, la azul el Y y la rosa el eje Z.

NOTA

Un vértice, cara u objeto, se vuelve de color rojo cuando está seleccionado y blanco cuando no lo está.

puede recuperar la vista original eligiendo la opción "Reset View" en el "Menú Emergente".

A continuación aprenderemos a seleccionar las partes del cubo que hemos creado. En la parte inferior de la zona de edición están las "Select Options" (opciones de selección). Podemos seleccionar vértices, caras o polígonos, grupos u objetos y uniones de huesos. Elegimos la opción "Select" (F1) en "Tools" y pulsamos en el botón "Vertex" para seleccionar los vértices inferiores del cubo. Nos situamos en la vista frontal y hacemos *clic* desplazando el ratón para abrir un área de selección que englobe los dos vértices inferiores. Una vez seleccionados se vuelven de color rojo. Observamos en la vista izquierda (Left) cómo se ha seleccionado sólo un vértice. (ver fig 5). Esto significa que al seleccionar en la vista frontal hemos elegido sólo los vértices situados delante; para elegir también los vértices traseros (los que no se ven) debemos deseleccionar la casilla "Ignore Backfaces" (ignorar caras traseras)

Para seleccionar una cara o polígono se realiza la misma operación, pero eligiendo la opción "Face". Hay que tener en cuenta que para seleccionar una cara es necesario elegir al menos dos vértices. Para elegir un objeto o grupo de objetos basta con pinchar sobre ellos con la opción "Group" elegida.

MOVER, ROTAR Y ESCALAR UNA PRIMITIVA

Vamos a mover nuestro cubo por la cuadrícula. Primero lo seleccionamos y luego pulsamos el botón "Move" (F2). Ahora es cuestión de hacer *clic* en cualquier punto de la vista deseada y desplazar el ratón con el botón izquierdo pulsado. Cuando elegimos ciertas herramientas como mover, rotar o simplemente crear un cilindro, aparecen en la parte inferior de la zona de herramientas los diálogos de opciones correspondientes.

Sabiendo esto, podemos también desplazar de una manera más exacta un objeto, y es introduciendo valores para los respectivos ejes y pulsando el botón "Move" de "Move Options". (ver. Fig 6).

Para rotar (F3), tenemos que tener en cuenta el punto de referencia donde girará el objeto. Para elegir este punto tenemos tres opciones: "Center of Mass" (centro del objeto), "Origin" (origen del objeto) y "User Point" (Punto por el usuario). Con la primera opción, se rota a partir del centro del objeto; con la segunda, desde su origen (determinada por la situación de los ejes); y con la tercera, a partir de donde hagas *clic* con el ratón.

Elijamos la opción "Center of Mass". Hacemos *clic* en la vista frontal y desplazamos el ratón con el botón izquierdo pulsado. Vemos cómo el objeto gira sobre su centro. Si pinchamos en la opción "Origin", el punto de rotación se colocará en el centro de los ejes X, Y y Z y el cubo pivotará sobre ellos. Podemos rotar con valores prefijados en las casillas numéricas situadas en "Move Options".

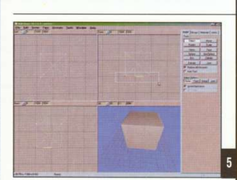
El procedimiento para cambiar de tamaño el cubo es similar a los anteriores. Pulsando el botón "Scale" (F4) podremos escalar el objeto a partir del punto de origen que marquemos en "Scale Options". (ver fig. 7).

Todas estas funciones de edición funcionan igualmente con cualquier selección que hagamos, ya sean vértices, caras u objetos. Es conveniente que practiques con estas opciones de edición y adquieras soltura en su manejo. Prueba con distintas primitivas, vértices o caras.



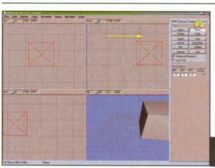
En el próximo número...

... aprenderemos todo lo necesario para empezar a modelar los objetos de *Zone of Fighters*.



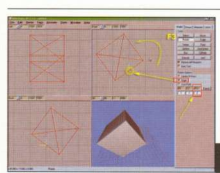
5

Selección de un vértice en la vista izquierda.



6

Pulsando el botón "Move", desplazamos el objeto.



7

Escalamos el objeto con el botón "Scale".

NOTA

La opción "Ignore Backfaces" permite seleccionar o no los vértices o caras que no se ven en una vista.

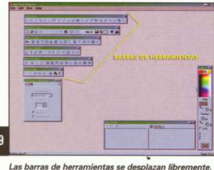
NOTA

***Milkshape3D* tiene hasta diez niveles de *undo* (deshacer) (CTRL + Z).**

MODELADO 2D

8

Vista del entorno de Paint Shop Pro.



9

Las barras de herramientas se desplazan libremente.



10

Empezando a trabajar: abrimos una imagen.

¿QUÉ ES *PAINT SHOP PRO* Y PARA QUÉ LO UTILIZAREMOS?

Paint Shop Pro es una aplicación para crear, editar y retocar imágenes. Gracias a su amigable interfaz, su precio y sus capacidades, se ha convertido en una buena alternativa a programas más populares, como *PhotoShop*.

Es ideal para dibujar todos los gráficos 2D que nuestro juego necesita como: pantallas de presentación, logotipos, iconos y bitmaps para sprites y texturas. Puede manejar cualquier tipo de formato de imagen conocido, trabajar con múltiples capas, crear y editar vectores y permite utilizar varios niveles de undo (hacer y deshacer acciones.)

PRIMERA TOMA DE CONTACTO

La filosofía de diseño de este programa se asemeja a la utilizada por Adobe *PhotoShop*, así que, si estás familiarizado con la aplicación de Adobe, no tendrás muchos problemas en habituarte a él.

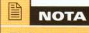
Al ejecutar *Paint Shop Pro* podemos observar varias paletas esparcidas por la pantalla y varias barras de herramientas, las cuales representan, mediante iconos, todas las opciones posibles del menú principal. Las barras de herramientas se pueden desplazar y colocar en el lugar de la pantalla que desees con el ratón; para ello, debemos mantener pulsado el botón izquierdo sobre las dos barras paralelas que se hallan en el comienzo de cada grupo de iconos mientras las desplazamos. (ver fig. 9). En la opción "ToolBars" del menú "View" podemos ocultar o mostrar estas barras de herramientas, así como las paletas. También pueden enrollarse y desenrollarse automáticamente, conmutando la acción en el botón de la flecha situado en la etiqueta.

Todas las opciones de cada herramienta se encuentran en la ventana "Tool Options".

EMPEZAR A TRABAJAR

Para comenzar a trabajar con *Paint Shop Pro* necesitamos tener una imagen. Podemos crearla nueva, cargar un archivo almacenado o importarla directamente desde un escáner o cámara digital. Para cargar una imagen almacenada elegimos la opción "Open" del menú "File"o pulsamos *CTRL + O*. Hay otra manera de seleccionar una imagen para cargarla y es utilizando la herramienta "Browse" pulsando *CTRL + B* o eligiendo la opción "Browse" en "File". Esta opción nos permite ver una miniatura de todas las imágenes que se encuentran en un directorio.

Basta con pulsar dos veces sobre la miniatura para cargarla directamente en el programa. Si lo que queremos es crear una imagen nueva, elegiremos "New" (CTRL + N). Posteriormente, aparecerá un cuadro de diálogo donde debemos elegir el tamaño, calidad y color del fondo. Una vez pulsado "OK" aparecerá una ventana nueva con el color que elegimos como fondo.



NOTA

El formato de imagen nativo del *Paint Shop Pro* tiene la extensión .PSP

TRUCO

Para ocultar o mostrar todas las paletas activas, pulsar la tecla de tabulación (TAB).Todas las opciones de cada herramienta se encuentran en la ventana "Tool Options".

En el próximo número...

... conoceremos las herramientas y nos prepararemos para empezar a realizar el logotipo de nuestro juego. Sólo con práctica, mucha práctica, dominarás *Paint Shop Pro 7*.

Herramientas para desarrollar el sonido de un juego

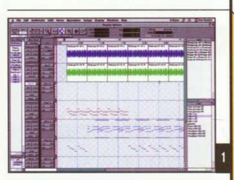
Antes de existir el sistema operativo Windows y los sistemas multimedia, la capacidad sonora en un PC de serie se limitaba a su altavoz interno. Lo más que podía generar un juego, por aquella época, era un montón de pitidos que, a veces, resultaban hasta molestos. Otros microordenadores habían conseguido con éxito reproducir sonido con cierta calidad, como en el caso del Amiga de Commodore que, prácticamente, basaba su publicidad en sus posibilidades multimedia.

Esta capacidad sonora no comenzó realmente en el PC hasta la llegada de las tarjetas de sonido, que proporcionaban al ordenador una expansión de su hadware de la que carecía. Los juegos empezaron a mostrar efectos especiales de sonido algo complejos y hasta música al mismo tiempo. El uso del PC para generar música sufrió un gran auge que, evidentemente, se extendió al sector lúdico. La necesidad de cubrir estas nuevas posibilidades provocó el nacimiento de numerosos softwares para la creación de música.

¿QUÉ PROGRAMAS SE UTILIZAN PARA HACER LA MÚSICA Y LOS EFECTOS DE SONIDO?

Por aquel entonces, antes de que aparecieran las tarjetas de sonido, sólo existían para PC interfaces MIDI para conectar el ordenador a instrumentos musicales. El ordenador proporcionaba, en este caso, secuenciadores que controlaban los eventos MIDI de los instrumentos. El más conocido fue el *CakeWalk*, que podía manejar hasta 256 pistas de eventos musicales simultáneamente. Las posibilidades musicales, en el ámbito profesional de aquellos tiempos, estaban en manos de los sistemas Apple y Atari. Los Macintosh de Apple acapararon toda la producción profesional de aplicaciones musicales con programas como *Perfomer*, *Pro Tools*, *MasterTrack* o *Visión*. Pero, para el usuario de a pie, estos sistemas eran económicamente inalcanzables, dando paso a los asequibles ordenadores Atari con interfaz MIDI incorporado y con aplicaciones realmente potentes como *Cubase*. Este programa se implementó posteriormente en los PCs, perpetuando así su existencia después de la desaparición de Atari y, hoy día, sigue siendo un estándar mundial en secuenciadores de mediano y alto nivel. Actualmente, con la versión 5 del *Cubase VST*, es posible manipular tanto MIDI como pistas de audio; forma parte, junto con otras aplicaciones como *Logic* o *Pro Tools*, del conjunto de herramientas más utilizadas para el empleo de instrumentos electrónicos en la creación de música para videojuegos.

Asimismo, el avance tecnológico en la generación de audio por ordenador y los nuevos formatos de audio han contribuido, en gran medida, al cambio en la utilización de sistemas para hacer música. Se han desarrollado aplicaciones para PC que emulan a la perfección cualquier generador de sonido externo como sintetizadores o cajas de ritmos. Ahora, por mucho menos dinero, es posible tener en tu ordenador más



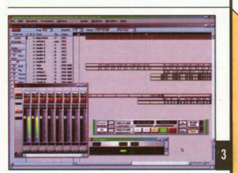
1

El programa ProTools fue el más popular entre los profesionales que usaban ordenadores Macintosh de Apple.



2

El Atari 1040 ST introdujo el uso del ordenador para la música en los hogares gracias a su precio y su interfaz MIDI incorporado.



3

Steinberg introdujo sus secuenciadores CUBASE en los Atari para uso amateur y profesional, y actualmente es uno de los más populares.

El programa LOGIC de Emagic es muy eficaz para la secuencia de pistas de audio a nivel profesional.



El programa Rebirth es un clásico mundial en cajas de ritmos virtuales para la realización de música de baile.



A diferencia de otros editores de audio, Cool Edit Pro ofrece la posibilidad de mezclar pistas de audio como lo haría un secuenciador.

DEFINICIÓN

▶ ¿QUÉ ES UN SAMPLER?

Samplers o muestreadores digitales son aparatos que permiten capturar con bastante exactitud cualquier tipo de sonido externo y convertirlo posteriormente en audio digital (sampler o muestra) para poder ser procesado.

prestaciones que instrumentos electrónicos de elevado precio. Igualmente, la posibilidad de introducir audio a través de la tarjeta de sonido desde fuentes externas como micrófonos o instrumentos electrónicos ha dado lugar a la utilización del ordenador como *sampler*.

Uniendo a esto la velocidad de procesamiento y la capacidad de almacenamiento de datos que proporciona un ordenador, tenemos un cóctel explosivo del que beben millones de usuarios profesionales y *amateurs*.

Aparecen en escena infinidad de aplicaciones para generar música y efectos de audio de manera cómoda y potente como: *Rebirth* o *Fruity Loops* ampliamente utilizados en la creación de música electrónica de baile, sintetizadores virtuales, procesadores de efectos, etc.

La posibilidad de almacenar sonido digital originó la aparición de sistemas para manipular esta información de audio. Ahora es muy común grabar en el ordenador (como si de una grabadora se tratase) audio procedente de cualquier fuente de sonido y poderlo editar y transformar posteriormente. Este tipo de operaciones se realiza con aplicaciones denominadas "editores de audio". Existe un gran número de estos programas, pero todos tienen en común la cualidad de poder visualizar, manipular y aplicar efectos a la onda sonora y de manejar los formatos más actuales de audio. Entre los más populares en el sector medio y profesional están *Sound Forge* o *Cool Edito Pro*. En nuestros días, es muy frecuente que cada tarjeta de sonido del mercado proporcione un programa de edición de audio. Incluso en Internet podemos encontrar este tipo de herramientas a precios realmente asequibles y en ocasiones gratis (opción que aprovecharemos para elegir las herramientas para hacer música y los efectos de sonido de nuestro juego).

¿QUÉ PROGRAMAS UTILIZAREMOS?

Como ya hemos diseñado, *Zone of Fighters* llevará música, efectos de sonido y voces. Para la música principal vamos a utilizar un secuenciador que nos permita mezclar pistas MIDI y pistas de audio al mismo tiempo. Podríamos elegir *Cubase* pero su precio es algo elevado, así que una buena opción son los programas shareware, muy asequibles y con bastante calidad. Encontramos uno que se ajusta perfectamente a nuestras necesidades, el *Anvil Studio*. Esta aplicación nos permitirá reproducir, grabar, componer y editar pistas usando ficheros de música en formato MIDI y pistas de audio en formato .WAV. También nos permitirá componer pistas rítmicas con nuestras propias muestras de sonido.

Otra forma de hacer la música para utilizarla en nuestro juego es por medio de secuencias de muestras. La mejor forma para realizarlo es usar un secuenciador tracker que nos permita utilizar el formato .X3M para almacenar nuestra música, y una buena opción es *FastTracker*.

Por último, necesitamos un editor de muestras para realizar los efectos especiales de sonido. Hemos elegido un conocido programa shareware muy fácil de usar, potente y con la posibilidad de manejar ficheros .MP3; nos referimos al *Goldwave*. Esta herramienta incorpora multitud de efectos para nuestras muestras, cambios de formatos y un sistema de generación de efectos personalizado por medio de fórmulas matemáticas.

⏭ En el próximo número...

... conoceremos las herramientas más utilizadas para componer el sonido de un videojuego y cuál utilizaremos nosotros para los efectos especiales y música de *Zone of Fighters*.

Conceptos básicos de Blitz 3D

Al igual que ocurre con el habla o con cualquier tipo de escritura, en la informática existen unas reglas determinadas que se deben seguir para su comprensión por otros individuos.

Un ordenador sólo entiende de ceros o unos, pero este sistema es muy difícil de comprender por el ser humano. Por este motivo, se han inventado multitud de lenguajes diferentes que interpretan estos ceros y unos para poder comunicarse con el ordenador de una forma más coherente y sencilla. Esta experiencia comenzó con el *lenguaje ensamblador*. A lo largo de la historia de la informática, estos lenguajes han aumentado en diversidad, utilización y potencia. Sin embargo, sólo algunos han obtenido la fama necesaria para prevalecer, ya sea por su facilidad de aprendizaje o por su uso a escala mundial. También es importante tener en cuenta a qué sector de la informática están destinados estos lenguajes dependiendo de su arquitectura. Así, por ejemplo, en Inteligencia Artificial dominaron el *Prolog* y el *Lisp;* en el aprendizaje de la programación tuvo las riendas el



Pascal, por su estructuración, y el lenguaje *C*, que actualmente es el más utilizado en todo el mundo, incluso para construir otros lenguajes; y, por último, destacar quizás el lenguaje de programación más popular de todos, debido a su sencillez de uso, el *Basic*.

Al margen de todos estos lenguajes principales o básicos nacieron otros muchos, consecuencia de las necesidades del mercado y del avance informático. Generalmente, son variaciones de sus abuelos o simplemente cambios en su sintaxis o actualizaciones de su vocabulario. A causa de este avance, casi todos buscaron sistemas visuales más cómodos y potentes para trabajar. Por ejemplo, el *C* evolucionó a un sistema de programación orientado a objetos. Por su parte, el primer *Basic* de PC se transformó en un sistema visual orientado a eventos para Windows.



El lenguaje *Basic* siempre ha sido el preferido para la gente deseosa de iniciarse en el mundo de la programación, ya que su sintaxis se acerca mucho al lenguaje coloquial y además permite un uso general. Con el auge de la imagen, el sonido y los videojuegos –en el ordenador personal–, aparecen en escena



multitud de nuevos lenguajes de programación dirigidos hacia la gente que quiere desarrollar aplicaciones multimedia y lúdicas *amateur* y semiprofesionales. La característica principal de estos nuevos lenguajes es su sencillez de aprendizaje y su relativa potencia. Ya en tiempos del Atari se dieron a conocer el *Stos*, el *Amos profesional* para ordenadores Amiga o el *Div Game Studio* para PC. También aparecieron aplicaciones programables para desarrollar juegos como *Klik and Play, 3D Construction Set* o el moderno *3D Game*

NOTA

Un programa básicamente está formado por una lista de instrucciones o comandos que el ordenador lee e interpreta secuencialmente dando lugar a imágenes o sonidos

NOTA

El compilador del *Blitz3D* es el encargado de transformar todo el listado de instrucciones escritas con el editor en código máquina, es decir, en el lenguaje que entiende el ordenador: ceros y unos



Studio. Actualmente, el número de nuevos lenguajes orientados a programar juegos y basados en la sintaxis del Basic es extensa. Aun así, *Blitz3D* destaca por encima de todos.

Blitz3D, sucesor de Blitz Basic, como su propio nombre indica, está basado en el *Basic*, aunque también lleva influencia de otros lenguajes como *C* o el antiguo *Amos*.

Para escribir un programa en *Blitz3D* (B3D) hay que tener en cuenta ciertas reglas gramaticales para que el intérprete del lenguaje pueda leerlo y pasarlo al compilador sin problemas. Antes de seguir, hay que aclarar que llamaremos "listado" al conjunto de instrucciones que forman el programa.

COMENTARIOS Y PALABRAS RESERVADAS

Generalmente, resulta útil el uso de tus propios comentarios en el listado de tu programa. Estos comentarios sólo tendrán valor explicativo en el conjunto de instrucciones y serán ignorados por el compilador. Para insertar comentarios en *B3D* se utiliza el símbolo ";". Se puede insertar cualquier tipo de símbolo como comentarios al principio de una línea de código o al final de sentencias, por ejemplo:

```
Function Visualizar_terreno()
....
End Function
Esto también es válido:
Function Visualizar_terreno();
Aquí empieza la función de
visualizar el terreno
....
End Function
```

Las palabras reservadas son aquellas que son utilizadas por el propio lenguaje y no pueden servir para definir algún tipo de dato por el programador. Por ejemplo, no podemos definir print=3 porque print es un comando del lenguaje para imprimir en pantalla caracteres. De todas formas, el editor indica cuándo escribes una palabra o signo reservado, ya que estos adquieren el color azul.

DATOS, IDENTIFICADORES Y TIPOS DE DATOS

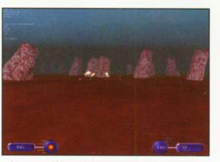
Los datos son las posiciones de la memoria del ordenador que contienen los valores que utiliza el programa. Para que el programa pueda utilizar estos valores almacenados en memoria, es necesario que a los datos se les asigne un nombre; ese nombre se denomina *identificador*.

Por medio de estos identificadores podemos dar nombre a variables, constantes, funciones, etc.

Hay ciertas reglas para definir un identificador. Siempre debe empezar por un carácter alfabético, aunque después esté precedido por números o símbolos; por ejemplo los siguientes identificadores son válidos:

```
Vidas
Vida_jugador
E_1
Explosion_1
M_
multi_jugador_1
```

Resulta indiferente utilizar mayúsculas y minúsculas, así *Vidas, vidas* o *VIDAS* es lo mismo. También *B3D* puede identificar nombres iguales para tipos de datos distintos; por ejemplo, si utilizas el nombre *Vida* para definir una variable y el mismo nombre para defi-



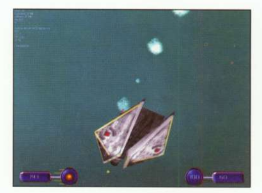
nir una función en el mismo programa, *B3D* sabrá en cualquier momento a qué te estas refiriendo.

Hasta ahora hemos mencionado los términos "variables", "constantes", etc. Pero ¿qué son?

Sencillamente, son tipos de datos que se comportan y almacenan la información de distinta manera. Básicamente, en *B3D* hay cuatro *tipos de datos* diferentes según su funcionamiento: las *constantes*, las *variables*, los *arrays* o *matrices* y las *estructuras*. Y tres tipos según la información que almacenan: valores numéricos *enteros*, numéricos *flotantes* y valores alfanuméricos o *texto*.

Los valores numéricos *enteros* o *integer (Int)* no tienen parte fraccionaria o decimales; por ejemplo, son números enteros el 3, el 23 o el -42. *B3D* soporta rangos enteros desde -2147483648 hasta +2147483647.

Los valores de *punto flotante* o *float* (*Float*) incluyen parte fraccionaria y nos ayudan a definir operaciones matemáticas con más precisión. El decimal se pone como punto; por ejemplo, 20.3, -2.5, .45 (0.45), son valores





numéricos de coma flotante. Un detalle a tener en cuenta es que este tipo de datos es más lento de procesar que el valor numérico entero.

Los valores *alfanuméricos* o *strings (Str)* son realmente cadenas de texto. Un dato alfanumérico puede contener cualquier tipo de carácter incluyendo los números (estos números serán tratados como caracteres y no como valores matemáticos) y van entre comillas. Varios ejemplos de valores alfanuméricos podrían ser: "Jugador 1", "FIN" o "Sector 23".

VARIABLES Y CONSTANTES

Las *variables* son los tipos de datos más simples. Son nombres o identificadores que el programador asigna a una porción de memoria donde almacenará valores numéricos o caracteres.

B3D sólo puede identificar automáticamente el tipo dato numérico entero, no obstante, necesita ser avisado en los tipos numéricos flotantes y alfanuméricos; para ello, es necesario añadir un símbolo a continuación del identificador: "$" para texto y "#" para coma flotante.



Por ejemplo, en la declaración *Vida=3, B3D* entiende que "vida" contiene un valor entero. Aunque se podría poner también como *Vida%=3*. Para definir el mensaje "Sector 23 abierto" sería: Mensaje$="Sector 23 abierto". Y un ejemplo de declaración numérica en coma flotante sería: X#=120.45

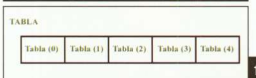
Estos símbolos sólo son necesarios la primera vez que se define el tipo de variable. No se puede elegir el mismo identificador para dos variables de distinto tipo, por ejemplo, es incorrecto elegir el identificador jugador$ y jugador%, ya que el primero sería una variable alfanumérica y el segundo una variable de tipo entero.

Para asignar un valor a una variable se utiliza el símbolo "="; por ejemplo, para asignar el valor *12* a la variable *puntos* se escribiría puntos=12 o puntos%=12 (Hemos reservado un sitio en la memoria con el nombre "puntos" y hemos guardado allí el valor numérico entero "12").

La denominación de variables viene determinada porque los valores que contienen pueden cambiar durante la ejecución del programa. Si definimos en el principio del listado la variable *vida=3*, estamos diciendo que el jugador tendrá al comenzar la partida tres vidas. Pero, en el caso de que pierda 1 durante el juego, esta variable debe cambiar su valor a 2; para realizar la resta basta con poner *vida=vida-1*. Esta misma operación nos servirá para cuando el jugador pierda otra vida más.

EL ALCANCE DE LAS VARIABLES

Al hablar de "alcance" nos referimos a hasta dónde puede llegar el valor contenido en una variable dentro del programa. *B3D* trabaja con



Array de una dimensión.

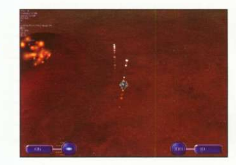
dos tipos de variables según su alcance: globales y locales.

Las variables globales tienen un alcance global y pueden ser usadas en cualquier parte del programa, es decir, que a esta posición de memoria definida como "global" se puede acceder desde cualquier sitio del programa. Sin embargo, las variables locales tienen su campo de acción en la función que las declara.

Para declarar variables de tipo global se utiliza la palabra *Global* antes del identificador, por ejemplo, *Global vidas=5*. Y para las de tipo local se utiliza *Local*, por ejemplo, *Local x=100*.

Si al declarar la variable no defines de qué tipo es, *B3D* entenderá que es una variable local.

Las constantes son en realidad variables pero con valores fijos que no serán cambiados durante la ejecución del programa. Adquieren utilidad en aspectos del programa que nunca cambiarían como el valor del seno de 90 o una resolución de pantalla. Para definir una constante se utiliza la palabra "Const"; por ejemplo, la variable constante más típica en un programa es PI, y se definiría al principio del programa como: Const PI= 3.141592





Varias constantes pueden ir separadas por coma en la misma declaración, por ejemplo:

```
Const largo_pantalla=800,
ancho_pantalla=600
```

En *B3D* existen dos constantes predefinidas y son: el valor de PI que es PI y los valores 1 y 0 como *True* y *False* respectivamente.

ARRAYS

Un array o matriz es en realidad un identificador que se relaciona con muchas posiciones de memoria. Estas matrices pueden ser de una o dos dimensiones. (ver figura 1)

Para definir un array se utiliza la palabra reservada Dim seguida del identificador y entre paréntesis la cantidad de celdas de memoria que se le asignará a ese identificador. Por ejemplo, vamos a definir una matriz unidimensional llamada "Enemigos" que contenga 5 celdas de memoria: Dim Tabla(4)

Os preguntaréis por qué 4 y no 5. Esto es debido a que la casilla primera empieza en el 0 y no en el 1, así pues tenemos 5 valores desde 0 hasta 4.

Podemos asignar valores a cada posición de memoria de la tabla en el momento de la declaración: `Dim Tabla(4)=12,3,4,60,100` esto significa que en la posición 0 de Tabla hay un 12, en la 1 de Tabla un 3 y así sucesivamente.

Para pasar un valor cualquiera de la Tabla a otra variable, por ejemplo el valor 60 a "fuerza", escribiríamos: fuerza=Tabla(3), porque la posición 3 (en realidad es la cuarta) contiene el valor 60.

Los arrays bidimensionales son como una matriz dentro de otra matriz. Por ejemplo, definimos la siguiente matriz bidimensional Dim posición_planta(4,2). Estamos definiendo 15 posiciones de memoria para la variable "posición_planta" (0..4 x 0..2 = 5 x 3), pero con un cierto orden. En realidad tenemos una cuadrícula de 5 filas por 3 columnas. En la posición 0 almacenamos 3 valores, en la posición 1 otros 3 y así sucesivamente. Para acceder a los valores o posición, en este caso podrían ser las coordenadas X, Y y Z, de la planta número 3 se escribiría (ver fig. 2):

```
X=posicion_planta(2,0)
Y=posicion_planta(2,1)
Z=posicion_planta(2,2)
```

En el siguiente número estudiaremos con más detenimiento cómo se usan las matrices en *B3D*

POSICIÓN_PLANTA

	X	Y	Z
posición_planta 0	(0,0)	(0,1)	(0,2)
posición_planta 1	(1,0)	(1,1)	(1,2)
posición_planta 2	(2,0)	(2,1)	(2,2)
posición_planta 3	(3,0)	(3,1)	(3,2)
posición_planta 4	(4,0)	(4,1)	(4,2)
posición_planta 5	(5,0)	(5,1)	(5,2)

2

Array de dos dimensiones.

DECLARACIÓN DE FUNCIONES

Las funciones son bloques de instrucciones que realizan una tarea determinada; por ejemplo, podemos tener una función para calcular e imprimir la puntuación del jugador en nuestro programa. De esta forma no tendremos que escribir los mismos comandos cada vez que queramos realizar estas operaciones. Las funciones pueden admitir y devolver valores, utilizándose en muchas ocasiones como procesos para calcular operaciones.



Para definir y abrir una función se utiliza la palabra reservada *Function* más el identificador y los parámetros y para cerrarla las palabras *End Function*. La estructura básica es la siguiente:

```
Function identificador
(parámetros)
   instrucciones
End Function
```

Los parámetros pueden ser una o varias variables separadas por comas que pasan a la función cuando es llamada y tiene que ser de tipo local. Para salir de una función antes de llegar a *End Function* podemos utilizar el comando *Return*, que si lleva detrás una expresión o una variable retornará también su valor.

Las funciones son una cualidad muy interesante del *B3D* y tremendamente útil, que seguramente inundará vuestros listados. No lo dejaremos aquí, ampliaremos el estudio de su uso en la siguiente entrega de esta obra.

En el próximo número...

... continuaremos aprendiendo a utilizar las instrucciones básicas de *Blitz3D*. Estudiaremos, entre otras cosas, qué son las sentencias, el verdadero motor de un programa.

Editor de niveles Q3Radiant (II)

En la primera parte de este tutorial aprendimos a movernos por el *Q3Radiant* y empezamos a crear una sala. Siguiendo con nuestro interior, vamos a añadir cuatro elementos cilíndricos que nos servirán como columnas en cada esquina de la habitación.

No podemos dibujar directamente en la parrilla primitivas como esferas, cilindros o *brushes* de distinto número de caras. Debemos dibujar primero un cubo, el cual contendrá cualquiera de los elementos anteriores. Por lo tanto, para nuestra columna haremos un cubo cerca de una de las esquinas de la sala, de 62 unidades de lado por 16 de altura para la base y luego crearemos el cilindro eligiendo la opción "Cylindre" en el menú "Curve". Recordad que hay que desplazar la base de la columna en la ventana de alturas y colocarla sobre el suelo.

Nuestra base no será cilíndrica sino que tendrá un pequeño chaflán en su parte superior para suavizar el filo. Este corte lo haremos con la función "clipper" . En primer lugar con *CTRL + TAB* elegimos la vista Frontal XZ, pulsamos en el botón "clipper" y con el cursor del ratón colocamos los dos puntos de corte como se muestra en la figura 1.

La parte del objeto que quedará después del corte se vuelve de color amarillo y el trozo que desaparecerá de rojo. Para culminar el corte pulsamos *Enter*.

Podemos ver en la ventana de cámara cómo el chaflán sólo está en uno de los lados. Para el filo que nos queda, cambiamos a la vista lateral (side) YZ. Colocamos los puntos de corte de la misma forma y pulsamos *Enter*. Antes de cortar definitivamente, se puede ver en la ventana de la cámara cómo quedará la figura; eso nos ayudará a no cometer errores. Si los puntos no los hemos colocado correctamente podemos anular la operación pulsando *Escape*. Cambiamos de nuevo a la vista superior y ya podemos mover la base de la columna a la esquina y le asignamos la textura de la piedra.
Seguimos con el cuerpo de la columna, que sí es cilíndrica. Sobre la base dibujamos otro cubo que llegue hasta el techo. A continuación, vamos a darle forma cilíndrica. Podemos crear varios tipos de cilindros dependiendo de la cantidad de polígonos. La más aceptable es la opción "Cylindre" del menú "Curve".

Bueno, ya tenemos nuestra columna. Para las otras tres sólo tenemos que duplicar la primera.

Seleccionamos desde la vista de cámara la base y el cuerpo de la columna. Para duplicarla podemos hacerlo pulsando *Espacio* o con *CTRL + C* (Copy brush) y *CTRL + V* (Paste Brush)

Una vez duplicada, la movemos hacia la otra esquina.



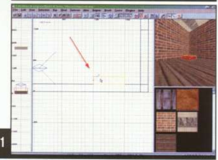
TRUCO

Para alejar o acercar la cuadrícula se pueden utilizar las teclas *Insert* y *Supr*.

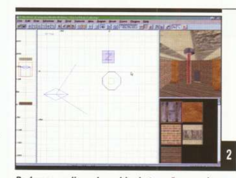


TRUCO

Podemos subir y bajar el objeto seleccionado con las teclas + y - del teclado numérico.


1

Con el cursor del ratón colocamos los dos puntos de corte.


2

Podemos realizar el cambio de tamaño usando el ratón.



NOTA

En la función "clipper", el trozo del objeto que corte dependerá de dónde se coloque el primer punto de corte. Si lo situamos más arriba que el segundo punto, cortará a partir de su derecha y si está colocado debajo cortará a su izquierda.

TRUCO

Podemos subir o bajar la cámara con las teclas "D" y "C".

Sólo veremos la luz cuando se compile del mapa.

Observamos que, para que encaje, tenemos que girarla 180 grados sobre su eje Z. Para realizar operaciones de rotación o inversión de ejes se utilizan los botones . Para nuestra columna elegiremos el botón . Vemos que cada vez que lo pulsamos gira el objeto seleccionado 45 grados sobre sí mismo (eje Z en vista superior).

Para el resto de las columnas realizaremos la misma operación.

FABRICANDO UNA LÁMPARA CON SU LUZ

Una vez que tenemos cada columna en su sitio vamos a colocar en el techo una simple lámpara con la luz encendida. Para el soporte de la lámpara colocaremos un *brush* de 8 lados.

Dibujamos un cubo pequeño y lo desplazamos hasta el techo; luego seleccionamos la opción "8 sided" en el menú "Brush". Para escalar el objeto libremente se utilizan los botones . Cuando pulsamos el botón de escalado , el objeto se torna de color verde. Desplazando el ratón haciendo *clic*, realizamos el cambio de tamaño (con *Escape* aceptamos la opción).

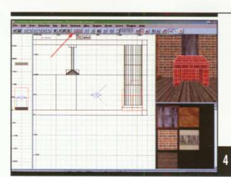
Para el tubo de la lámpara podemos copiar la base, alargarla y estrecharla. Para duplicar la base no utilizaremos *Espacio*, porque este procedimiento realiza un pequeño desplazamiento. Utilizaremos *CTRL+C* y *CTRL+V*. Una vez duplicada la base, movemos la copia hacia abajo con la tecla del *menos*, la alargamos hacia abajo y la estrechamos con la opción de escalado.

Para fabricar la cabeza de la lámpara vamos a utilizar un cono. Dibujamos el cubo y seleccionamos "Cone" en el menú "Curve" o bien elegimos la primitiva "Cone" en el menú "Brush" y así podemos darle la resolución que queramos.

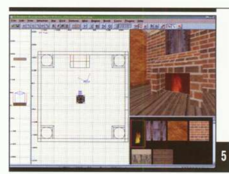
Una vez fabricada la lámpara, vamos a colocar la luz. Para ello, debemos convertir un pequeño cubo en una luz. Dibujamos el cubo en el centro de la lámpara; con el objeto seleccionado pulsamos el botón derecho del ratón y emergerá un menú. Elegimos la opción "Light" y luego pulsamos *Escape*. Vemos que el cubo se ha transformado en un rombo de color verde. En el editor no podemos apreciar la acción de esa luz. Sólo la veremos cuando se compile el mapa y sea renderizado por el motor del *Quake III*.

Vamos a terminar nuestra sala colocando una chimenea con fuego. Dibujamos un cubo en la pared y le realizamos un gran chaflán en su parte superior frontal. A continuación, le colocamos la salida de humo y posteriormente con la función "CSG Subtract" le practicamos el hueco donde irá el fuego. El hueco se hará dibujando dentro de la chimenea un cubo con el tamaño del hueco; pulsamos en "CSG Subtract" y borramos con la tecla *BackSpace*. Para el fuego podemos utilizar un "shaders" como efecto de llama. Debemos sumar una lista de shaders llamada "sfx" a nuestras texturas eligiéndolo en "Load from List" de "Load" en el menú "Textures".

La brillantez de tus niveles dependerá del modelado, diseño y calidad de las texturas que emplees. Este pequeño tutorial ha dejado al lado estas cualidades artísticas y se ha centrado en mostrar los pasos técnicos para comenzar a conocer este estupendo editor BSP.



Dibujamos una chimenea con su respectivo hueco para el fuego.



En el menú "Textures" podemos elegir la opción shaders.

TRUCO

Para desplazar la cámara libremente, nos situamos en la ventana de ésta y movemos el ratón con el botón derecho pulsado.

TRUCO

Si tenemos problemas en ajustar los objetos, siempre podemos reducir la resolución de la rejilla con las teclas del 1 al 7 del teclado no numérico.

En el próximo número...

... arrancaremos con una serie dedicada al sonido. Y la mejor forma de hacerlo es con el secuenciador más popular del mercado: *CUBASE VST*.

Los primeros ordenadores domésticos

Paralelamente al desarrollo de las consolas para juegos, en los 80 aparecieron multitud de microordenadores que proporcionaban una nueva forma de entretenimiento.

Estos sistemas dieron paso al desarrollo de nuevos géneros de videojuegos, con gráficos, sonidos y desarrollos más variados. A excepción de Commodore, prácticamente todos los ordenadores domésticos se basaban en los microprocesadores *Z80* y *Z80-A* de Zilog, de 8 bits de potencia. Destacaron los modelos ZX-Spectrum, MSX y Amstrad. Al principio, estos equipos mostraban sus posibilidades a través de juegos que se regalaban al comprarlos, generalmente versiones de recreativas de la época. A pesar de que el único avance que manifestaban estos ordenadores en muchos años era la ampliación de memoria, los juegos sí mejoraban mes a mes. El sector lúdico empezó a crecer de nuevo gracias al surgimiento de numerosas desarrolladoras y distribuidoras de juegos. Esto ocurrió gracias a la arquitectura abierta de estos sistemas, que permitía a cualquiera que tuviera conocimientos de *BASIC* o código máquina programar su propio juego y conseguirle una distribución. Muchas de las compañías de juegos que hoy conocemos nacieron en esa época. Otros sistemas como los PC basados en procesadores Intel tenían todavía un precio demasiado alto y su utilización para jugar no estaba muy extendida.

Los 8 bits realmente dominaron en el mundo de la informática personal desde los años 80 hasta la década de los 90.

MSX

Este tipo de microordenador se usó hasta casi 1995. MSX nace de la iniciativa de varios fabricantes que buscaban una norma común que permitiese fabricar ordenadores compatibles entre sí. Eran equipos basados en los procesadores Z80A a 3,5 Mhz. Tenían la particularidad de poder ser ampliados mediante cartuchos. Poseían de 16 hasta 64 Kb de memoria para los MSX-1 y hasta 256 Kb para los MSX-2, además de un teclado semiprofesional. Llegaron a ser muy populares, debido también a la calidad multimedia: incorporaban sonido de 3 canales y 8 octavas y gráficos de hasta 256 x 192 con 16 colores a la vez en pantalla en los modelos MSX-1 y 512 x 512 píxels y 256 colores en los modelos MSX-2.

No fueron muy populares en Europa y EEUU debido a un escaso marketing. Aun así, los MSX hicieron furor en sitios tan dispares como Japón o Brasil. Aunque la calidad de sus juegos era muy superior a la de sus competidores de 8 bits, no tuvieron mucho éxito comercial ya que los mejores los incluían en cartuchos, encareciendo con ello el producto.

La posibilidad de asignar hasta 16 colores diferentes a un solo píxel de pantalla, y los sonidos en varios canales, posibilitaban unos juegos realmente atractivos. Desarrolladoras de recreativas como Konami fueron las que más títulos realizaron para



1

Los primeros ordenadores de 8 bits y algunos ejemplos de cómo eran sus juegos en el comienzo.



2

Los ordenadores MSX y MSX2 llegaron a ser muy populares en algunos países.



3

Los juegos para MSX tenían una calidad extraordinaria y fueron muy abundantes.



El Commodore 64 y VIC-20 triunfaron en Estados Unidos y eran muy avanzados para la época.



Algunas pantallas de juegos para Commodore 64, sin duda una máquina adelantada a su época.



Los ordenadores de 8 bits de Amstrad tuvieron mucho éxito en Europa.

NOTA

El Commodore 64 hizo furor en EE.UU., mientras que en Europa los usuarios se decantaron más por el ZX-Spectrum.

MSX. Para MSX-1 destacan juegos como *Zaxxon, Ahtletic Land, Nemesis, Knightmare, Blade Lords* o *Akin*. Los sistemas MSX-2 llegaron a competir con los Commodore Amiga con juegos tan fantásticos como *Super Deform Snatcher* (exclusivamente vendido en Japón) o el *Pennant Race* 1 y 2, juegos que necesitaban la friolera de 64 Kb para funcionar. No podemos olvidar el *Metal Gear* 1 y 2, antecesores del actual *Metal Gear Solid* para PlayStation 2.

COMMODORE VIC-20 Y 64

Commodore VIC-20 surgió en el 1980 y revolucionó el mundo de los juegos para ordenador en EEUU. Ya por entonces disponía de un procesador independiente para el manejo del vídeo. Se llegaron a vender un millón de unidades y se realizaron más de 500 juegos en cassette para él. Evolucionó hasta el Commodore 64, sin duda una máquina adelantada a su época (1982). Basado en el microprocesador 6510 de 8 bits, disponía de hasta 64 Kb de memoria y una resolución de 320 x 200 a 16 colores. Pero lo que hizo potente a este ordenador no fue el escaso megahertzio de velocidad de su micro sino la separación de tareas que proporcionaba su arquitectura, ya que poseía un procesador 6567 que controlaba hasta 64.000 píxels (puntos de pantalla) de vídeo y podía generar 64 *sprites* a la vez y un procesador 6581 para controlar el sonido.

Todos los juegos del momento eran versionados para este ordenador y tenían una calidad envidiable para aquellos años. El éxito del C64 se centró en EEUU. En el mercado europeo lo tuvo más difícil, debido al dominio del ZX-Spectrum. Además, las nuevas desarrolladoras se decantaban por el Z80 por su facilidad de programación.

Hasta que no apareció el Amstrad CPC, los reyes del sector eran los Spectrum y C64.

AMSTRAD CPC

Este ordenador nació en 1984, de manos de Alan Sugar, para competir con el ZX-Spectrum. Estaba basado también en el microprocesador Z80, y tenía incorporado un cassette para cargar y grabar los programas; el equipo completo incorporaba un monitor en color de 14 pulgadas. Gozaba de mayor memoria (64 Kb) y de la posibilidad de utilizar un color por cada punto gráfico de pantalla, lo cual permitía desarrollar juegos visualmente más ricos.

Al igual que C-64, los CPC incluían sendos procesadores para controlar el vídeo y el audio por separado.

Casi todos los juegos realizados para Spectrum y MSX-1 fueron versionados sin dificultad para Amstrad, ya que todos tenían en común el mismo procesador.

Este ordenador evolucionó en el CPC-6128 con 128 Kb que sustituyó la unidad de cassette por la de disco. Poco a poco fue desplazando al Spectrum de Sinclair, cuya compañía terminó comprada por Amstrad, quien siguió comercializando los Spectrum 128 +.

NOTA

Actualmente son muy populares entre los "nostálgicos" los emuladores de los primeros ordenadores domésticos. En el CD-ROM adjuntamos algunos.

En el próximo número...

... hablaremos de la introducción del ZX Spectrum, de su época dorada y del avance tecnológico de sus juegos.

Cuestionario Videojuegos



Preguntas

1. **¿Cuáles son los tipos de datos según la información que almacenan?**

2. **Un edificio tiene 10 pisos y 20 habitaciones por cada piso.**
A. Define una tabla para saber la posición exacta de cualquier habitación.
B. Introduce en la variable *"habitación"* la habitación 15 del piso 3

3. **¿Qué se describe en un guión técnico?**

4. **¿Cómo debe ser el comportamiento de los personajes secundarios de un juego para ganar jugabilidad?**

5. **¿En qué consiste modelar en baja poligonización?**

6. **¿Qué es el "Browser" del *Paint Shop Pro* y cómo se activa?**

7. **¿Qué es un sampler?**

8. **Enumera al menos tres secuenciadores para ordenador .**

9. **¿Qué procedimiento debemos seguir para añadir una geometría que no sea cúbica en el *Q3Radiant*?**

10. **¿Cómo podemos añadir una luz en el *Q3Radiant*?**

Respuestas al cuestionario 2

▷ **1.** **a.** F5 **b.** CTRL+ F4 **c.** CTRL + H **d.** CTRL + INICIO / CTRL + FIN
e. INICIO / FIN **f.** CTRL + CURSOR DCHO / CTRL. + CURSOR IZQ
g. CTRL + C **h.** CTRL + V **i.** CTRL + F **j.** CTRL + R

▷ **2.** Se pulsa CTRL + R para ir a la ventana de "Reemplazar". En "Buscar" se escribe la palabra que se quiere cambiar y en "Reemplazar por" la palabra nueva. Luego se pulsa en "Reemplazar todo".

▷ **3.** En un sistema 3D se puede ver la acción desde distintos ángulos en tiempo real, mientras que en dos dimensiones sólo dispondremos de una vista.

▷ **4.** El rendimiento en un videojuego consiste en la cantidad de imágenes por segundo que se consigue. Su importancia radica en mantener esa velocidad constante y superior a 24 frames (imágenes) por segundo para evitar saltos en la acción.

▷ **5.** Para el diseño gráfico 2D, el *Photoshop* de Adobe. Para el modelado y animación en tres dimensiones, el *3D Studio Max* de Discreet.

▷ **6.** En primer lugar se realiza un diseño en papel mediante un boceto, a continuación se construye, modelándolo en un programa 3D. Una vez modelado, se crean las texturas en un programa de dibujo o bien se texturiza directamente en una aplicación especial como *Deep Paint 3D*. Por último, queda definir los movimientos que tendrá el modelo en un programa de animación.

▷ **7.** Generalmente se utilizan los formatos .WAV, .MP3, .X3M, y .MIDI para la música y el formato .WAV, de nuevo, para los efectos especiales de sonido y las voces en off.

▷ **8.** El formato .WAV ocupa mucho espacio en el disco y en memoria, sin embargo, almacena mayor calidad de audio y al no estar comprimido su reproducción es inmediata. Por otro lado, el .MP3 ocupa hasta 14 veces menos capacidad y mantiene una calidad muy similar al formato .WAV, pero tiene el gran inconveniente de que al estar comprimido necesita ser descomprimido en tiempo de reproducción, disminuyendo el rendimiento del juego.

▷ **9.** Las texturas pueden estar en formato TGA de 24 bits de color pero sin canal alfa, o a 32 bits con canal alfa. También se pueden utilizar texturas en formato JPG sólo a 24 bits de color.

▷ **10.** Esta herramienta se denomina "Surface Inspector" (inspector de superficies) y la activaremos pulsando la tecla "S".
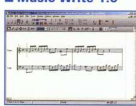
Contenido CD-ROM 3

▶ AUDIO

■ S-Cal 4.1

Calculadora de b.p.m, tiempos de silencio, tiempos de retraso, etc... Muy útil para darle la forma final a nuestra melodía.

■ Music Write 1.0



Herramienta de composición musical muy intuitiva y que incluye multitud de efectos y prestaciones.

■ DubIt 2.0.1

Útil aplicación multimedia que permite integrar sonidos y música en formato .wav a imágenes o clips de vídeo de un modo muy cómodo.

■ AcidPro 3.0

Potente herramienta de creación musical, con una gran capacidad de composición y edición.

■ Acoustic Labs Mixer 2.1

Podemos convertir fácilmente nuestro PC en un editor de audio y en una mesa de mezclas gracias a este programa.

■ Digital Vision DSP 151

Procesador de sonido digital en tiempo real.

■ Cakewalk

Para grabar, reproducir y organizar nuestra música digital con una completísima herramienta.

DISEÑO 2D:

■ Texture Processor 1.3



Podremos crear variadas texturas para nuestras imágenes de un modo sencillo.

■ Object Paint 1.0

Sencilla aplicación que nos será de gran ayuda cuando queramos dibujar formas básicas rápidamente.

■ Fractal Explorer 1.23

Gracias a este programa, podremos crear fractales para realizar los fondos del juego.

■ PictureViewer 1.0.54

Útil herramienta que nos dará una vista rápida de las imágenes que tengamos almacenadas.

■ Neat Image 1.1

Corrector de imágenes y fotografías, para que éstas se vean libres de efectos y brillos indeseados.

■ ThumbsPlus

Programa para localizar, ver y catalogar nuestras imágenes.

▶ DISEÑO 3D:

■ It'sMe 2.0



Divertidísima herramienta con la cual podremos modelar personajes 3D con fotografías de nuestros amigos.

■ CyberMotion 3D-Designer 7.0

Modelado, animación y renderizado de objetos 3D.

■ 3D Reducer 1.1

Optimización de modelos 3D con visualización de éstos en tiempo real.

■ DesignWorkshop Lite 1.8.4

Completo programa que nos proveerá de todo cuanto necesitemos para crear y grabar nuestros propios modelos en 3D.

■ ModelMagic 3D

Creación de objetos 3D a base de Open GL.

■ Tutorial texturas Q3 Radiant

Tutorial del Q3 Radiant.

■ Milkshape 3d 1.5.10

Modelador 3D en baja poligonización y fácil de usar.

▶ PROGRAMACIÓN:

■ Game Maker 4.1



Completo paquete destinado a que personas sin conocimientos previos de programación puedan crear un juego desde cero.

■ Digital Mars C/C++ Compiler 8.28

Potente compilador de C y C++ para poder crear programas en entornos Windows.

■ Borland C++ Compiler 5.5

Rápido y optimizado compilador para el lenguaje C++ de 32 Bits.

■ FreePiXCL 4.48

Desarrollo de aplicaciones multimedia de un modo rápido y sencillo.

■ Xtreme Diagram++ MFC Library - MFC 7.0 Compliant 4.20

Permite al desarrollador añadir interfaces gráficas a las aplicaciones Windows.

▶ EMULADORES Y JUEGOS:

Emuladores:

■ WinFrotz 5.3

Emulador de los clásicos juegos de DOS.

■ Mega Drive Emulator 0.99a



Con este emulador podremos convertir nuestro PC en una consola virtual Sega Genesis.

Juegos:

■ Death Zone 1.0

Divertido juego de tipo Arcade con 20 niveles en los que matar malvados aliens.

■ Classic Basic Games 1.0

Conjunto de juegos clásicos de las décadas de los 70 y 80.

■ Zone of fighters

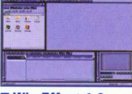
Nueva entrega de nuestro juego Zone of fighters, con muchas mejoras.

▶ VÍDEO:

■ Moviedb 1.3.1.0

Organizador de películas y clips, para que siempre sepamos dónde tenemos cada cosa.

■ MovieXone 4.0.



Una de las herramientas de creación y edición de vídeo más potentes y populares.

■ Win Effect 1.0

Con este programa podremos crear interesantes efectos de vídeo.